Evolving a Bridge Bidding System

by

Jason R. Woolever

B.S., Massachusetts Institute of Technology (2000)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2001

Author .... [illegible]
Department of Electrical Engineering and Computer Science
May 25, 2001

Certified by .... [illegible]
Una-May O'Reilly
Research Scientist
Thesis Supervisor

Accepted by .... [illegible]
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Evolving a Bridge Bidding System

by

Jason R. Woolever

Submitted to the Department of Electrical Engineering and Computer Science
on May 25, 2001, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

In this thesis, I designed and implemented WEBB, a computer program that produces Bridge bidding systems. WEBB uses a genetic algorithm which leverages expert Bridge knowledge to search the space of Bridge bidding systems. One way in which WEBB differs from other (computerized) Bridge programs is that it learns a bidding system rather than, typically, using a statistically-based search of bidding and card play options or using a naive, pre-scripted convention/system. To learn better and better systems, WEBB uses a co-evolutionary approach. Each system in a population competes with one another in tournaments to acquire IMPs. IMPs endow a system with breeding potential, allowing genetically superior bidding strategies to populate ongoing generations. A bidding system is represented procedurally, i.e. as a policy. Thus, WEBB conducts a search, co-evolutionary, in policy space.

Thesis Supervisor: Una-May O'Reilly
Title: Research Scientist

Acknowledgements

I would like to thank my parents, Richard and Beverly Woolever. Without their support over many years, this work would have been impossible. Second, I would like to thank Dr. Una-May O'Reilly, whose encouragement and direction helped me bring this work to maturity. Many students do not have an opportunity to define and execute their own research, so I greatly appreciate her support. Finally, I would like to thank Qixiang Sun for introducing me to Bridge and for numerous conversations that have helped guide my work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

During the mid-1990s, AI research groups developing machine players for various games scored wins against world champions in the corresponding domains. The first instance was CHINOOK beating Don Lafferty at Checkers in January 1995 [10]. Next came Deep Blue over Garry Kasparov at Chess in May 1997 followed by Logistello over Takeshi Murakami at Othello in August 1997 [2, 8]. More difficult games such as Go retain human world champions, but this could easily change as computers become more powerful, as Go's higher branching factor may be overcome with more advanced computers.

Games of imperfect information fall into a different category, however. Matthew Ginsberg's GIB, a program that plays (i.e. bidding and card-play) Bridge, attempts to overcome this obstacle by performing large Monte Carlo simulations to score possible actions [6, 7]. However, even after performing many tens to hundreds of simulations per decision, GIB performs well below the expert level at Bridge bidding. Inherent in converting an imperfect information problem to that of solving many problems with perfect information are two major defects, noted by Ginsberg:

> 1) ... the approach never suggests making an "information gathering play."
>
> 2) ... it is weak at ... combining ... possibilities.

No additional amount of computer power will overcome these deficiencies, so other approaches must be considered.

Bridge can be divided into two components: bidding and play. While full machine players need to perform both tasks, it is often useful to consider each process independently to gain insights. While play is interesting in its own right as a game of imperfect information and deduction, it is well studied with machine players performing near or at the expert level even with the limitations noted above.

As a result, this work is focused on creating a machine bidder. This problem can be further divided into two components. The first is using bids to construct a language, known as a bidding system, as a framework for communication between players. The second is choosing bids allowed by the language which are best suited for situations as they come up. The latter is an interesting problem of searching and evaluation techniques, but this research targets the root of the problem, the language itself.

1.1 Problem History

Initial attempts at machine bidding used rules-based approaches, operating entirely within the imperfect information domain (unlike GIB). Two early systems were created by G. Carley in 1962 and A. Wasserman in 1970 [3, 4, 11]. These approaches likely resulted from the overall lack of processing power necessary for exhaustive searches, but indicated that such systems could fare well against average players at the bidding portion of Bridge. These systems were fundamentally limited by the person who encoded the rules, however, and more importantly, by the rules themselves.

In 1983, E.T. Lindelöf released information on COBRA, a computer optimized bidding system generated by statistically examining several million boards [4, 9]. Using these deals, Lindelöf created rules for calculating a *basic points count* (BPC) and a *distribution points count* (DPC) from hand features and known features of partner and opponents. Another rules based system assigns meanings to bids, providing information about features, causing BPC and DPC to be recalculated, and ultimately using them to determine contracts.

COBRA performed well in constructive auctions, but performed poorly amidst opponent interference, partly because of the disruption of the flow of information between the players, but also because COBRA was not equipped to use the information revealed by the opponents by their disruptions. While COBRA's rules system resulted from extensive statistical analysis covering card layouts better than hand-crafted rules, planning for all possible types of interference by opponents was impractical.

1.2 Challenges

In addition to interference, it is important to understand other difficulties specific to the Bridge domain before constructing a bidding model.

1.2.1 Partnership Interaction

The first challenge, a corollary to interference, is partnership interaction. Even if the opponents are silent in an auction, traditional planning techniques applied to generating a sequence of bids to best describe a hand will often fail because it is difficult or impossible to anticipate how the bidder's partner will respond to intermediate bids. This may create a situation where the remainder of a plan being executed is no longer valid.

One possible approach to this problem is to disregard planning techniques and use a greedy strategy, choosing the best bid in each situation without consideration of later potential difficulties. Clearly some tradeoff is required to minimize awkward bidding situations while exchanging useful information.

1.2.2 Partial Information

In addition to other problems that are a consequence of Bridge being a game of partial information, certain difficulties arise as a more direct result. A first approximation of the effective bidding would focus on efficiently exchanging information between partnerships. However, any information that a player reveals also helps the opponents evaluate the current situation. Thus it may be a better strategy for players to reveal less information which is more relevant to their partners rather than the most complete and accurate information possible. This brings up a related point: psychic bids.

1.2.3 Psyching

Due to the varying utility information has to different players, it is sometimes useful to choose a bid that misrepresents the hand of the bidder. Psyches can be minor, representing a slight deviation from partnership agreements, while others can derail the bidding for the opponents or hopelessly mislead partners. A common psyche is to show strength and length in a suit, when the bidder has none. This is especially true in the third-seat position, when the opponent yet to bid rates to have a strong hand. The psychic bid can cause the opponents to never consider choosing that suit as trump and can cause them to form a grossly inaccurate perception of card distribution. Even if the psyche is discovered later in the auction, the disruption could cause irreparable harm. If the partner of the psyching player has length in the suit instead, s/he may try to compete in that suit, expecting substantial trump support and overall strength, yet receive none, resulting in a large penalty.

Choosing when to psyche and learning how to identify psychic bids are not easy tasks. However, the odds tend to be against the psychic bidders because the invalid information tends to mislead partners more-so than opponents. Implementing psyches in a computer bidding program would greatly complicate the overall model to produce a small effect. Therefore, the model described in the rest of this document never generates psychic bids and has no provisions for identifying and responding to them.

1.2.4 Evaluation

Evaluating the fitness of a bidding system accurately can be very difficult because its performance is strongly dependent upon the types of bidding systems it competes against. One way to deal with this problem is to maintain a sufficiently large and diverse population of bidding systems to rate an unknown system against. This metric, however, is computationally expensive and dependent on the diversity of the population.

Additionally, playing only one board against each member of the population may not be sufficient because systems that are better across all boards may underperform on a random sampling. This is a common issue across many games. Nevertheless, as a result, it is not possible to tell with certainty which bidding system is best from a set of candidates, but a good approximation is sufficient for most purposes.

1.3 Approach

Bridge bidding itself is commonly divided into two domains: constructive bidding and competitive bidding. The former involves trying to bid accurately to games and slams while the latter involves shutting out or safely out-bidding the opponents. While competitive bidding is present, WEBB (Woolever's Evolving Bridge Bidder) focuses on the former. No special considerations are added for effective competitive bidding, but could be added at a later time.

Rather than trying to construct a complicated model that accounts for the difficulties mentioned, WEBB consists of a general, yet robust, knowledge-based framework that an evolutionary algorithm can operate on to evolve strategic, policy knowledge related to bidding. Expert knowledge is leveraged in the design of the framework so that the algorithm can exploit it to evolve bidding system structures that decide what test to make and what to do as a result of the test.

The evolutionary algorithm allows diverse strategies to be exercised and evaluated by having them compete with each other. In WEBB, play takes place as tournaments among the population. IMPs are awarded (or subtracted) depending on the performance of systems versus each other. This is feedback to the evolutionary adaptation process as a fitness value. The best systems survive and adapt, through natural selection and genetic inheritance. Offline, periodically, the system with highest IMP total is evaluated by playing it against itself on 1,000 hands and versus others from other runs and/or other epochs.

WEBB gains its power not from either the expert knowledge in the model nor the genetic algorithm, but the fusion of both into one system. The abstractions made by the infused knowledge allow the evolutionary algorithm to search the 'right' space, i.e. that of bidding system procedural knowledge, while the evolutionary algorithm, using co-evolution in the form of tournament play, provides a tractable way to explore the possible solution space.

Chapter 2 provides an overview of WEBB describing hand evaluation and introducing bidding system design. Chapter 3 describes the bidding system framework in more detail, demonstrating how bid generation and bid explanation are performed using a common data structure. Chapter 4 reviews learning techniques WEBB applies to produce bidding systems and what evaluation techniques are used. Chapter 5 summarizes the results of WEBB trial runs and compares WEBB's performance to that of GIB and human experts. Finally, Chapter 6 discusses what compromises are made in WEBB's design and how it might be improved. A glossary of Bridge terms is provided in Appendix A, as well as a brief discussion of duplicate Bridge in Appendix B.

Chapter 2

Model Description

Since the solution space for this problem is so large, using modules to enforce abstractions simplifies evolution, making the process more efficient and the results easier to understand. If the abstractions are too restrictive, however, a good solution may not fit the model representation because of the inherent complexity in an effective solution. Some abstractions can be included in the representation itself, allowing the evolutionary algorithm to reject or modify its use. This is not always possible, however, so some hard-coded abstractions are necessary. In this case, abstractions should be carefully chosen from the tried-and-true abstractions common to most human-designed bidding systems. This endangers the precept that human influence is to be avoided, however. To minimize this effect, every such abstraction must be scrutinized and justified. For each module in the model, whenever there is an abstraction being made, a justification is provided for it.

2.1 Hand Evaluator

One such abstraction is to separate hand evaluation and bid description. Human Bridge players rarely look at a Bridge hand as just a set of thirteen specific cards. Rather they generalize by counting the number of cards in each suit, by noting specific cards such as aces and face cards, and by noting the placement of these keycards (i.e. a king supported by an ace or a queen is much more valuable than a singleton king).

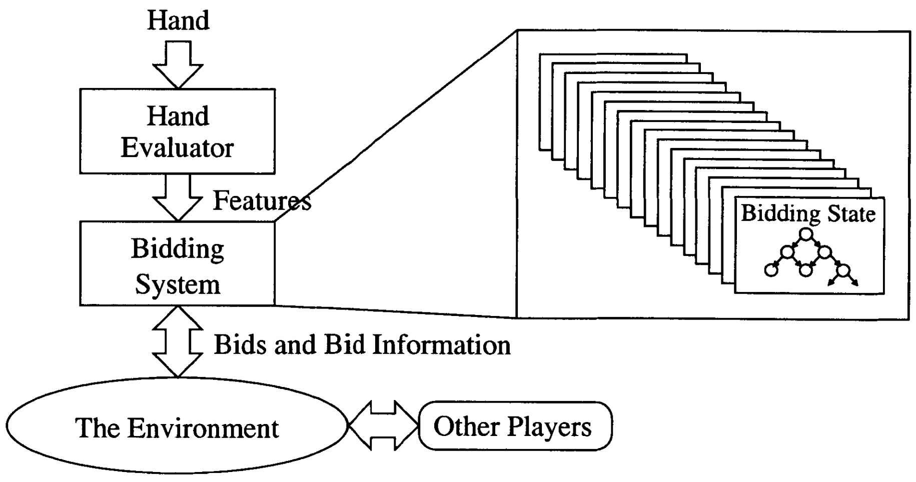


Figure 2-1: WEBB overview

The purpose of WEBB's hand evaluator is to generate useful generalizations about hands to aid the other components.

While it is possible to model each bidding situation as a complicated function mapping individual cards and knowledge of other players to bids, WEBB more practically performs several less-complicated calculations once to generate higher-level features of a hand and then operates on those features (via the bidding system module) instead to generate bids. Occasionally bids may rely on information about individual keycards, such as aces, kings, and sometimes queens, but usually only to refine information already revealed by higher-level features. This abstraction is made in all known bidding systems to date. WEBB does not have any notion of keycards, but adding it would be an easy task. WEBB, at present, calculates strength and uses suit-length (sections 2.1.1 and 2.1.2).

2.1.1 Strength

All other things being equal, the value of an ace or a card of any rank can be estimated by correlating the expected number of tricks that can be won in no-trump or a suit contract versus the count of cards of that rank. For example, if a hand with one king tends to produce one more trick than a hand with no kings, a king could be given a weight of 1/13 the total of all cards. Many bidding systems today use the Milton Work count, weighing each ace as four high-card points (HCP), each king as three HCP, each queen as two HCP, and each jack as one HCP. Thus, the expected HCP of a hand is ten. These values are easy to remember, but most experts agree that aces and tens should be valued slightly higher (than four HCP and zero HCP respectfully) and that queens and jacks should be valued slightly less.

Using double-dummy results, the appropriate values can be calculated exactly, assuming double-dummy play. This assumption will work well for systems generated with WEBB, because contract evaluation assumes double-dummy play, but the values may not be perfectly accurate for human play. One might expect queens, for example, to be undervalued using double-dummy results, because declarers often face a guess of which opponent has a queen of some suit. With double-dummy play, the declarer will never guess wrong, and queens will win fewer tricks. This difference is minor, however, and will not adversely affect the results.

	Milton	COBRA[1]	WEBB
Ace	4.000	4.048	4.280
King	3.000	2.857	2.740
Queen	2.000	1.905	1.544
Jack	1.000	0.952	0.822
10	—	0.238	0.371
9	—	—	0.156
8	—	—	0.069
7	—	—	0.017

Table 2.1: HCP values for various models

In WEBB, HCP are computed to the nearest 5/64 (0.078), so some value is also given to sevens, eights, and nines. WEBB derived its HCP values using the method described above, over Matthew Ginsberg's library of double-dummy deals. An ace, on average, produces 1.391 tricks, so it has a HCP value of $1.391 \cdot 40/13 = 4.280$. Other values are computed in a similar fashion.

[1]BPC, adjusted so that the expected hand point count equals ten

2.1.2 Shape

Other key features about a hand are characteristics of its shape. The simplest such features are the numbers of cards held in each suit in a hand. Higher-level features can be represented as compositions of suit lengths to determine whether a hand is balanced, semi-balanced, or one, two, or three-suited, etc. However, such features can be composed of lower-level suit-length descriptions, and thus do not need receive special consideration in WEBB.

2.2 Bidding System Descriptor

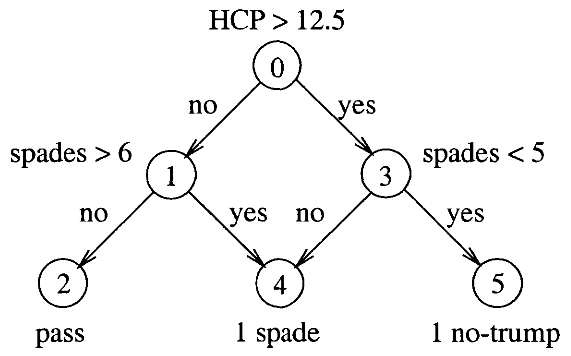
This module is the heart of WEBB. Its first task, as the *bid selector*, is to choose a bid given a hand and a bidding history. After selecting a bid, the bidding system has a second task, as the *bid descriptor*, to provide other players with a list of features that can be inferred from the bid. These two functions are, therefore, intricately linked, and need to be considered jointly at all times. Consider if this was not the case, allowing the bid selector to be an arbitrary function, represented as a black-box. The best the bid descriptor could do is apply the black box to various sets of random, but plausible, inputs to attempt to generalize the behavior of the box. Not only would this be very difficult to implement, but it would be computationally expensive and produce poor results.

If the bid selector is modeled instead as partitioning the hand space in the context of the bidding history, the bid chosen will correspond to some subset of hands defined by one or more partitions. The bid descriptor thus has the easier task of describing the subset of possible hands by inspecting the relevant partitions. This is the method used by WEBB.

At the top level, each bidding system is modeled as a large finite-state machine (FSM). Each bidding state has a program to select a bid and maps each possible bid to a new state, describing the transitions. Each partnership uses the same system, and hence the same FSM. Thus, when a bid is chosen with a corresponding transition to a new state, the bidder's partner will choose a bid based on the program contained in the new state. Each partnership begins in state 0. The number of states, as well as the length of the program in each state, are parameters that may be adjusted.

Bid selection is therefore execution of the program in a bidding state. To perform bid description, it is necessary to analyze the program. It is convenient to think of the program as a decision graph, where each intermediate node corresponds to a conditional branch, and each leaf node corresponds to a bid chosen. The bid descriptor then has the task of describing the path(s) from the root node to a node that corresponds to the selected bid.

Thinking back in terms of partitioning the hand space, the decision graph represents each partition with a decision node. Figure 2-2 shows a sample program and the corresponding decision graph. The two-dimensional variable space is shown, with the partitions defined by the decision nodes. In an actual example, there may be many more variables, producing more dimensions, but the effect is the same.



```
0: if my HCP > 12.5 skip 2
1: if my spades > 6 skip 2
2: pass
3: if my spades < 5 skip 1
4: bid 1 spade
5: bid 1 no-trump
```

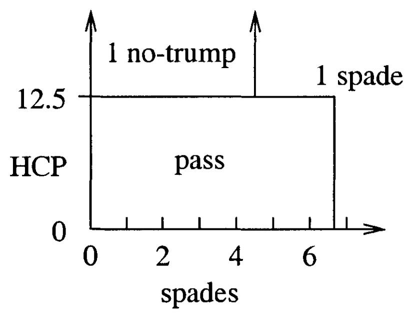


Figure 2-2: Bidding state program, decision graph, and partitioned hand space

Chapter 3 explains how the program-decision graph duality can be exploited for bid description in more detail, which is key to WEBB's performance. Decision and bid nodes are discussed in more detail, including implementations of the nodetypes used in WEBB.

Chapter 3

Bidding System Detail

Chapter 2 introduced a WEBB bidding system as a FSM with a program in each state to select a bid. Each program can also be interpreted as a decision graph with decision nodes and bid nodes. This chapter digs deeper into these data structures and describes algorithms for operating on them for selecting and describing bids.

3.1 Bidding States

In addition to the program contained in each bidding state used to select a bid, each state must also list transitions to other states, corresponding to all possible bids. If in the process of selecting a bid, control would run off the end of the program, it may continue execution in some other program in a different bidding state instead. In this manner, bidding states can use the information contained in other bidding states. To prevent infinite loops, if this occurs more than a fixed number times (high enough for state inheritance to be useful, but low enough to not hinder performance), the player is assumed to choose 'pass' instead. Each entry in the program is 32 bits, specifying a decision if the most significant bit is one or a bid if the bit is zero (refer to figure 3-1).

If randomly generated, each program within each bidding state will create a decision node at the root. Otherwise the program always chooses the same bid, a behavior that tends not to be effective. If an unconditional bid is desired, it is still possible to do this with either a mutation or having the root decision node be a decision that

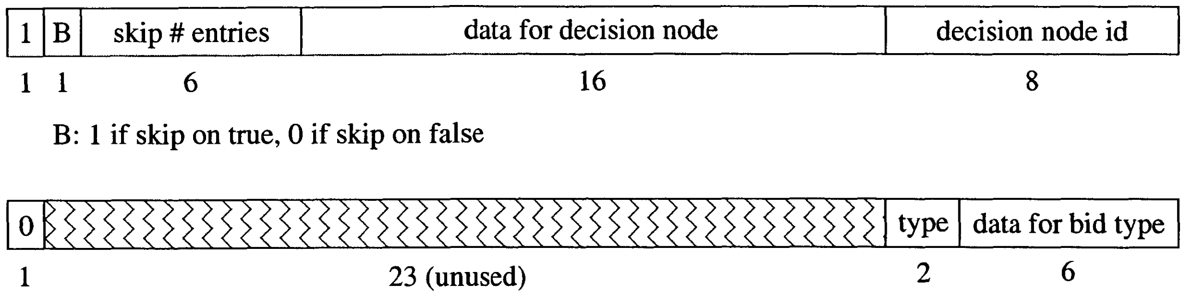


Figure 3-1: Bidding state program entry

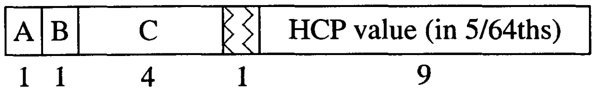
always produces the same result.

3.2 Decision Nodetypes

Each decision node uses 24 bits to describe what the decision is to be made at a given node (see figure 3-1). The least significant eight bits select which type of decision and the remaining 16 bits are data, forming one or more parameters to specify constants. Each decision nodetype used is described in more detail below. While WEBB currently uses far fewer than the 256 possible nodetypes, the added capacity allows additional nodetypes to be added with ease and chosen with different probabilities.

3.2.1 Strength

The strength decision type compares the HCP total of one or more players to a constant and returns true if the sum is always at least (or at most) the specified constant. In the case of just looking at the bidder ($B = 1$ or $C = 1000$), a simple comparison against the holding of the player is sufficient. However, whenever other players are involved, the minimum (or maximum) HCP holding of each player needs to be used. Next, because the HCP of all players must sum to 40 (by Bridge convention), if the players not involved have at most (at least) $40 - \mathit{constant}$ HCP, the affected players must have at least (at most) *constant* HCP. For example, consider the case of a decision node that branches if partner and LHO have 20 or more HCP. If they have are known to have at least 20 HCP, or the remaining players (RHO and the bidder in this case) have at most 20 HCP, the branch will be taken. Thus, if the branch is not taken, and RHO is known to have at most 12 HCP, the bidder has revealed that s/he has at least 8 HCP. Otherwise the bidder could deduce that the condition must be true. This type of deduction is built into WEBB.

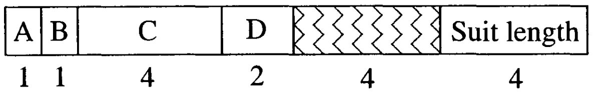


A: 0 if at most, 1 if at least
B: 0 if only me, otherwise consult C
C: sum me, pard, LHO, RHO
(0000 and 1111 become 1100)

Figure 3-2: Strength decision type

3.2.2 Suit Length

The suit length decision type works in very much the same way as the strength decision type. In this case, the constant is only four bits because there are only 13 cards in each suit. There is an additional parameter to specify which suit to include. Because each player has exactly 13 cards, WEBB can make deductions about other suit lengths, given information about the others. For example, if a player reveals having four clubs, four diamonds, and at most three hearts, the player must have at least four spades.



A: 0 if at most, 1 if at least
B: 0 if only me, otherwise consult C
C: sum me, pard, LHO, RHO
(0000 and 1111 become 1100)
D: suit

Figure 3-3: Suit length decision type

3.3 Bid Nodetypes

Bid instructions can be one of four types, each returning a Bridge bid in the context of an auction. Each takes a six-bit number as a parameter (refer to figure 3-1).

direct This instruction maps 0 to 1♣, 1 to 1♢, ...34 to 7NT, 35 to 1♣,... and 63 to 6♠. If the bid is invalid, it returns no bid.

relative This instruction bids a suit at some level above the current level, based on bits 3–4. Bits 0–2 select the suit (0 and 5 to clubs, 1 and 6 to diamonds, 2 and 7 to hearts, 3 to spades, and 4 to no-trump). Bit 5 is unused. For example, 000100 corresponds to a no-trump bid at the lowest level and 001010 corresponds to a jump-shift in hearts.

steps This instruction uses only bits 0-2, assigning 0 to the cheapest available bid (excluding pass unless a free bid is available), 1 to the next available bid, etc.

pass\[re]double This instruction uses only bit 0. If the bit is zero, [re]double is selected (or no bid if invalid). If the bit is one, pass is selected.

3.4 Bid Description

The above infrastructure describes how a player chooses a bid, but this is not sufficient to implement a bidding system. There must be a way for other players to determine features of the bidder's hand consistent with the bid(s) made. Using the above data structure, it is relatively easy to determine these features. One procedure determines features of individual bids and another makes inferences across multiple bids.

3.4.1 Feature Inferencing Algorithm (FIA)

This algorithm constructs a directed graph from the bidding program and walks over it, pruning unreachable nodes. If there is only one path through the graph to reach the chosen bid, all nodes will be pruned not in that path. Similarly, if the bidder made a decision that observers cannot determine, the reduced graph will have multiple paths to the chosen bid, making it more difficult to determine features.

Beginning at the bottom of the bid selection program, each instruction is examined. In each case a node is added to the graph. For this reason, it is efficient to implement as an array with as many slots as program entries. Depending on what the instruction is, the algorithm responds differently:

Bid, not chosen Since the bid was not selected, the bidder did not take a path to this node. Thus, insert it into the graph, but as a deactive node.

Bid, chosen Upon finding an instruction that produces the chosen bid, insert it into the graph as an active node.

Conditional, both children active With both children having a path to the chosen bid, either branch could have been chosen. Add a node to the graph and connect the children appropriately, adding constraints on both outgoing arcs corresponding to the decision that was made. Copy the constraints from each child onto the arcs and verify that an impossible result has not been reached. An example of this would be that a player holds fewer than five and more than four spades. In this event, trim the arc and revert to the one-child case. If both arcs have common constraints, these correspond to features that are known if that node is traversed, regardless of which way the decision was made. As such, copy it onto the node itself, to be further pulled up the graph toward the root.

Conditional, one child active With only one child active, if the decision was taken, it is certain which way it was made. Add a node to the graph, setting the appropriate constraint on the node itself. Ordinarily you would set the node to have only one child, but to create a smaller graph, it is possible to adopt any constraints on the child and add arcs to its children, effectively bypassing the child node. Inductively, this operation collapses all chains of nodes into single arcs with the intersection of the constraints on the chains. If a contradiction is ever reached, trim the arc and revert to the no-children case.

Conditional, no children active With no active children, the common parent was not traversed. Add it to the graph, but as a deactive node.

Because of the constraint annotation process, any features that are known from the bid will be constraints on the root node, where they may be simply read off. There may be additional information available later, however, if there are still decision nodes in the graph with two children. Information that is inferred later can create a contradiction on one of the arcs, indicating that the player chose the other path in the prior round. Thus, after constructing the graph in this manner, copy any remaining active nodes and connecting arcs into a repository in an attempt to extract more information later.

3.4.2 Graph Interference Algorithm (GIA)

Once two or more graphs are available, both graphs have inferred information at the root. This information can be applied to all arcs and nodes in other graphs. To do so apply this information to the root of the other graphs and push it down the graph. Then beginning from the leaves and walking back up the graph, if the new information creates any contradictions, prune the affected arc(s) and perform the compaction steps described in FIA. If this ultimately creates more certain information at the node, this process may be repeated on other graphs.

In theory, it may be possible to prune additional paths based on information contained deeper within each graph, but this event is rare and computationally difficult to detect. Doing so will not be useful anyway, unless it causes cascading contradictions, yielding more information at the root.

Chapter 4

Learning Through Co-evolutionary Tournament Play

To develop intelligent bidding systems, WEBB creates a population of random bidding systems then uses an evolutionary algorithm to adapt towards improved bidding systems. There are several parameters to set:

Population seeding whether initial systems are initialized with random bits, with previously evolved systems, or with winners of tournaments of random micro-populations to boost initial population fitness

Population count the number of populations to simulate in tandem

Population size the number of systems in each population, generally held constant throughout a simulation

Bidding state count the number of bidding states per member bidding system, generally fixed across all bidding systems

Program entry count the number of entries in each bidding state. This value is constant across all bidding states

Inheritance count the number of times control is allowed to flow off the end of a program to that of another bidding state before a choice of 'pass' is assumed

Fitness threshold IMP total required of each member to avoid replacement

Push penalty intensity of the penalty applied to systems that are too similar to other population members

Migration frequency if multiple populations are used, the probability that a bidding system is replaced with a migrant from another population

The population is tested and adapts towards improved performance on a library of bridge deals. WEBB uses Ginsberg's double-dummy database, including over 700,000 hands. 1,000 hands are reserved as an evaluation set that the best systems periodically bid to evaluate performance. Comparing each result with the double-dummy par is a convenient way to estimate fitness, but this result does not have a direct result on ordinary Bridge play.

Using the deal library, WEBB holds a round-robin tournament among the population members. In each matching, each system plays a board twice, once in each direction. The net IMP result is added or subtracted from the fitness of each system. If the IMP total ever falls below the fitness threshold, the system is replaced and the resulting IMP deficit is spread over the population to maintain a zero IMP population sum. Once each system has competed against each other system, the population is sorted by fitness, so that the best members can be tested or examined. Then a new round-robin tournament is held, repeating the process. As poor systems are replaced, the population presumably becomes more competent at bidding.

4.1 Replacement Operators

With the exception of migration, when a bidding system is replaced, it is replaced with either a mutated variant of another system or with some crossover of two parent systems. Some evolutionary operators produce subtle changes, suggesting hill-climbing within a local region of the space of bidding systems. Other evolutionary operators are more aggressive, tending to produce lethal mutations, but are responsible for leaps in the evolution process.

4.1.1 Mutations

Each mutation copies one parent making one or more small changes. These changes can be any of the following types.

Bit Mutation This mutation changes each bit in each bidding state with some low probability.

Entry Mutation This mutation replaces each entry in each bidding state with a random entry with some medium probability.

Entry Insertion This mutation inserts a random instruction into one bidding state, moving all instructions after the instruction down one slot (discarding the last instruction). Each decision node before the affected instruction has its branch target incremented if it points to an instruction after the insertion. Each decision node after the affected instruction has its branch target decremented if its branch target is in an inherited program. These modifications preserve as many arcs in decision graphs as possible.

Entry Deletion This mutation deletes an instruction in one bidding state, moving all instructions after the instruction up one slot (replacing the last instruction with a random entry). Each decision node before the affected instruction has its branch target decremented if it points to an instruction after the insertion. Each decision node after the affected instruction has its branch target incremented if its branch target is in an inherited program. These modifications preserve as many arcs in decision graphs as possible.

Transition Mutation This mutation replaces each FSM transition with some medium probability.

4.1.2 Crossovers

These operators attempt to take useful traits of two systems and combine them into one.

Bitwise Uniform Crossover This operator copies the parents bit-for-bit, choosing randomly with equal probability to choose each bit from either parent. If the parents are very dissimilar, this operator will almost always be fatal, but population homogeneity allows it to produce useful offspring.

Entry Uniform Crossover This operator copies the parents at an entry-for-entry level within each bidding state, choosing randomly with equal probability to choose each entry from either parent. Transition tables are similarly chosen randomly between the parents.

State Uniform Crossover This operator copies entire states from the parents, choosing randomly with equal probability to choose each state from either parent. FSM transitions remain attached to the state and are inherited accordingly.

State Adoption This operator copies a master parent system in its entirety except for one state which is chosen randomly from the other parent. State uniform crossover only allows state n to be crossed to state n. This operator relaxes that constraint.

4.1.3 Migration

Finally, if multiple populations are being run in parallel, some replacements can copy the best members of other population into the vacancy. Alternatively, if the best bidding systems are occasionally saved, this method could also try reintroducing an old bidding system into the population.

4.2 Population Control

Using the replacement scheme introduced above, certain safeguards are necessary to ensure population diversity and turnover. Running as described, especially among smaller populations, all member systems will eventually converge to some behavioral equivalent. At this point replacement will halt, as all tournament matches will be pushes, producing no net IMP score, preventing fitness scores from falling below the replacement threshold. To combat this, a scoring adjustment is made for systems which obtain push results consistently.

The penalty applied is zero for the first push, but increases linearly for all subsequent pushes. When the system produces a non-zero result, its push count is reset to zero. All IMPs deducted in this manner are distributed to all population members uniformly to maintain a zero IMP population sum.

This process has two effects. First, if a population starts becoming too homogeneous, the penalties applied will increase, resulting in an increased amount of replacement, promoting diversity at the macro level. Second, systems that use strategies that are different from the rest of the population will be less affected by the penalty, effectively gaining IMPs when the collected push penalty IMPs are redistributed uniformly. If the difference is sufficiently bad, the system will still be replaced, but if the difference is neutral this bonus prevents the system from being promptly replaced, promoting diversity at the micro level.

4.3 Evaluation

While the described system is sufficient for learning, it lacks an objective way to evaluate performance. Because systems are only compared to each other, with no absolute fitness function, it is difficult to rate the effectiveness of systems across populations and across all times. While it might be possible to score a bidding system against a suite of benchmark bidding systems, the result is highly dependent on the benchmark suite chosen because bidding systems can sometimes interact in complicated ways. For example, system dominance is not transitive, so on a given test set of boards, System A may beat System B which beats System C which beats System A.

Instead, a more objective way to evaluate fitness is to compute the *double-dummy par* for a board and to compare the performance of a system played against itself to the par score. The par result is that which is obtained if both pairs bid perfectly (and in the case of double-dummy scoring, omnisciently). The difference (converted to IMPs) will therefore be zero for perfect bidding and nonzero if one side or the other could have acted differently to get a better result for its side. While Bridge players are not omniscient causing double-dummy par results to not always correspond to observed par results from actual play, the measurement is approximate and objective.

However, even after computing the average IMP difference, the metric is not completely accurate. If bidding systems were evolved to minimize this value, they would learn to work cooperatively to get results near this magic number. Instead, bidding systems are competitive and are willing to increase the variance of the results if the expected result is good for themselves. A good example of this is the use of preempts which do exactly that. This limitation can have a significant effect and is discussed in more detail in the next chapter.

However, the metric is still a useful measure. For the 1,000 board evaluation set, the average IMP difference for using a pass-always strategy is 8.4. The experimental runs in the next chapter were able to produce strategies that reduce this to 5.5 IMPs, on average.

Chapter 5

Experiment Observations

To best observe population dynamics and the learning process, initially two single populations were simulated, without migration. The first had a population size of 128 and the second had a population size of 256. Both populations were initialized randomly, had four bidding states with eight instructions each, allowed three occurrences of inheritance, required each member to maintain -15 IMPs and had push penalty of $P/256$ IMPs (with P = number of sequential pushes). Next 13 runs were created to be run in tandem with migration. After a considerable amount of time, the migration rate was increased tenfold to see if it would affect the run. The runs are summarized in table 5.1.

5.1 Run A (Single population of size 128)

Operating on the 1,000 test boards, a bidding system using an always-pass strategy would obtain a score of 8.400 using the objective function described in the last chapter. After playing 500,000 boards, the population of 128 was able to score 5.700 IMPs from par on average. An additional 600,000 boards reduced this to 5.507 IMPs. The running time was approximately four days on a Celeron 550MHz computer.

	Run A	Run B	Run C
Population seeding	Random		
Population count	1		13
Population size	128	256	128
Bidding state count	4		8
Program entry count	8		16
Inheritance count	3		
Fitness threshold	-15		
Push penalty	$P/256$		
Migration frequency	n/a		1/1000[1]
Total boards played	$1.73 \cdot 10^8$	$1.75 \cdot 10^8$	$4.99 \cdot 10^9$
Objective Func. Min.	5.507	5.762	5.495

Table 5.1: Run parameter summary

Each member played an average of 102 boards before being replaced. If each member has approximately the same probability of being replaced after each board, a Poisson distribution describes the number of boards played by each system and the expected age of the oldest member at any time should be approximately eight times the average age (816 boards) for a population of size 128. In the steady state, this is the case. In instances where a favorable mutation or crossover allows a new member to perform well relative to the rest of the population, it can take thousands of iterations before the rest of the population catches up and eliminates the member.

Initial systems learned gradually, advancing from 7.5 to 7.4 IMPs from par until 33,000 boards into the simulation when the system shaved off a full IMP by learning to bid 3NT cooperatively. One player opens 2♢ with 14.3+ HCP and partner bids 3♡ with 4.5+ HCP support, followed by a forced 3♠ and 3NT. After 290,000 boards per member, the pattern simply follows 1NT–3NT, with 1NT replacing 2♢ as 14.3+ HCP and 3NT as 7.9+ HCP. Players also try to compete in spades holding 6+.

When a mutation reduced the number of spades necessary to compete from six to five, the objective function improved further. However, such aggressive bidding is dangerous, and indeed, WEBB evolves to double more freely leading to a large

[1]Increased to 1/100 at time $2.5 \cdot 10^7$ boards per member ($3.20 \cdot 10^9$ boards total)

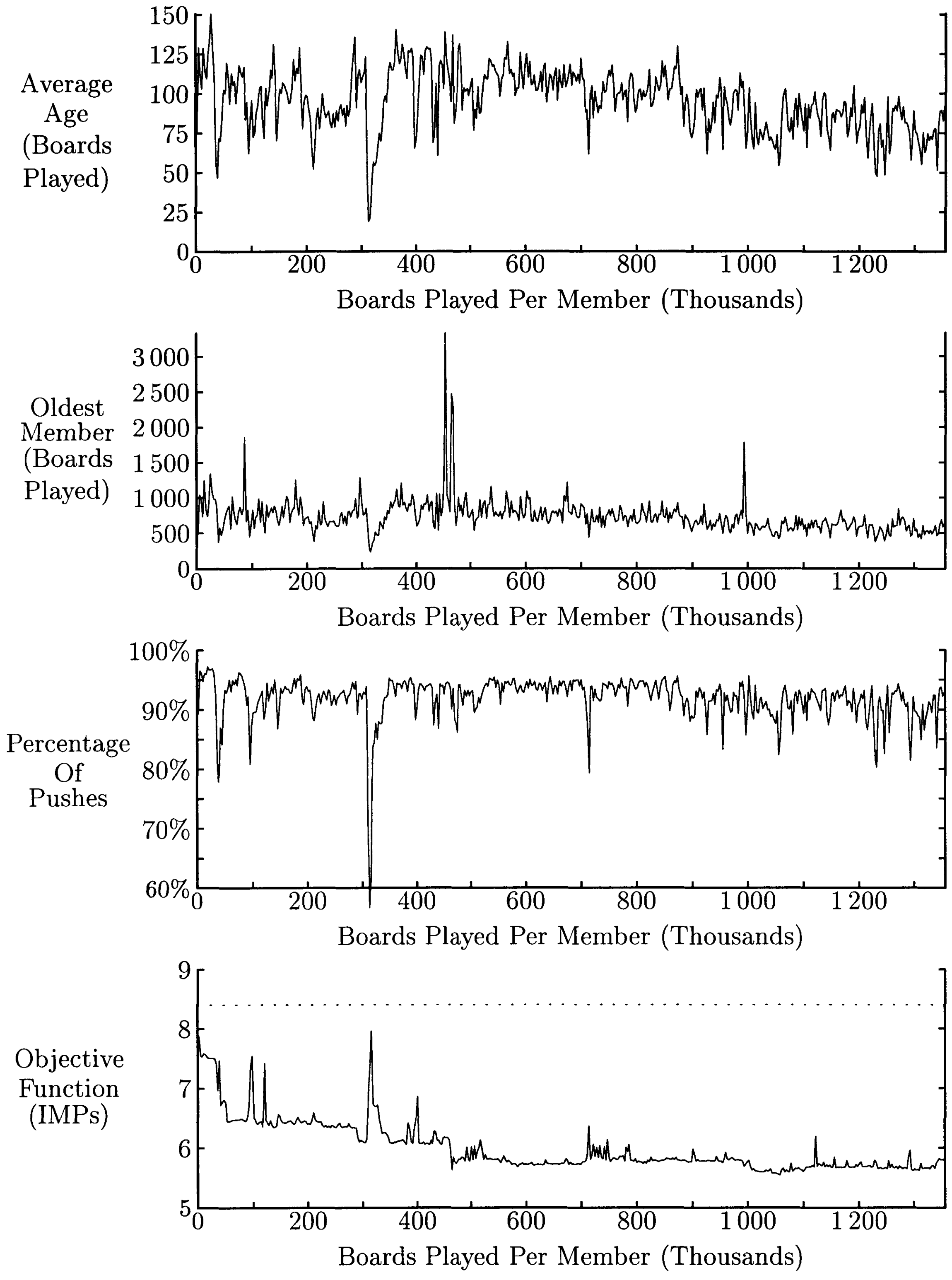


Figure 5-1: Run A (Single population of size 128)

spike in the objective function after 310,000 boards per member. Correspondingly, average age, maximum age, and push percentage plummet. After 30,000 more boards per member, the disruption is over, with the best system still bidding spades naturally holding five, but doing so only at lower levels. Eventually the system achieves slightly better results by bidding spade and heart games. Slams are never investigated intelligently.

Dlr: East	♠ AK	
Vul: N/S	♡ Q876	
	◇ 4	
	♣ AKQ1072	
♠ 10		♠ J987642
♡ A932		♡ 5
◇ J10652		◇ A9
♣ 963		♣ J84
	♠ Q53	
	♡ KJ104	
	◇ KQ873	
	♣ 5	

West	North	East	South
		1♠[1]	Pass[2]
Pass[2]	1NT[3]	2♠	3NT[4]
Pass	Pass	Pass	

3NT by North, making 5
N/S Score: 660 (Par = 660)

(1) 6+ spades and 0–14.2 HCP
(2) 0–5 spades and 0–14.2 HCP
(3) 14.3+ HCP
(4) 7.9+ HCP

Figure 5-2: Run A example board

5.2 Run B (Single population of size 256)

The larger population run was very similar to the smaller run, but did not advance as far. It was only able to reach 5.762 IMPs from par given the same amount of CPU time (nearly 700,000 boards per member) compared to 5.507 for the smaller run. Once again, the early population only made slight advances until learning to bid 3NT cooperatively, but did so much later than in the smaller run. The best system evolved to find spade games aggressively.

Both populations exhibited some interesting traits. Most significant improvements in the objective function are preceded by slight increases. These events can be described by the introduction of a superior bidding system into the population which has not yet played enough boards to have the highest fitness. In the meantime, because

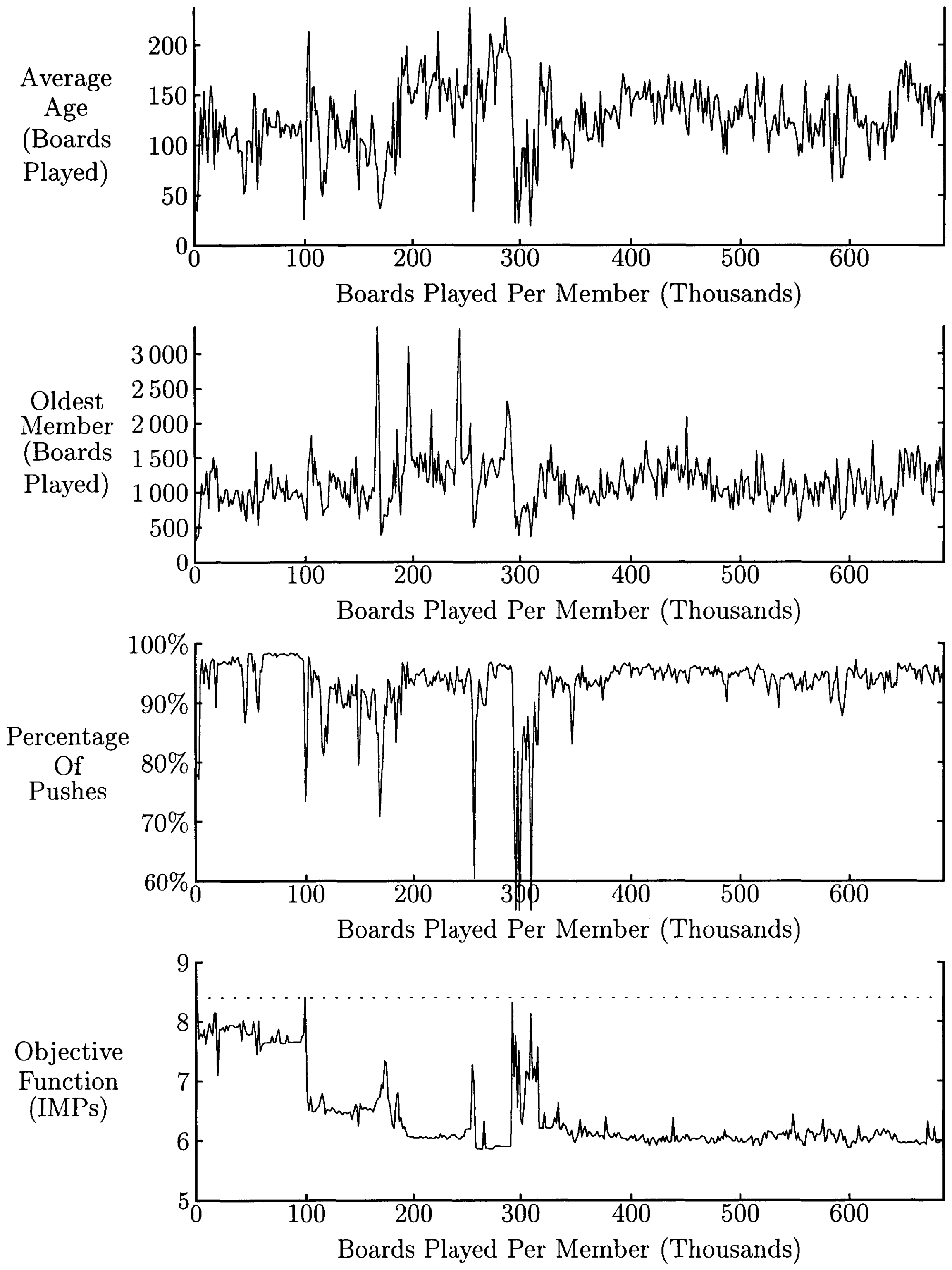


Figure 5-3: Run B (Single population of size 256)

it uses different strategies than other bidding systems, the previous best system may be demoted while a lesser system is lucky and gets promoted. As a result, between the time the superior system is introduced into the population and the time it takes for it to play enough boards, an inferior system may be temporarily ranked first.

```
Dlr: North  ♠ QJ952
Vul: N/S    ♡ K8
            ♢ Q8
            ♣ 9753
♠ 7                     ♠ K864
♡ QJ102                 ♡ 653
♢ K97                   ♢ AJ62
♣ QJ642                 ♣ 108
            ♠ A103
            ♡ A974
            ♢ 10543
            ♣ AK
```

West	North	East	South
	1♠[1]	Pass[2]	3♢[3]
Pass	4♠	Pass	Pass
Pass			

4♠ by North, making 4
N/S Score: 620 (Par = 620)

(1) 5+ spades and 0–13.0 HCP
(2) 0–13.0 HCP
(3) 0–11 clubs and 12.5–19.0 HCP

Figure 5-4: Run B example board

5.3 Run C (Multiple populations of size 128)

Introduction of multiple population with migration introduced produces interesting effects. Superimposing the objective functions of the 13 population leaders (figure 5-5), its behavior becomes a bit more clear. Some of the populations seem to make dramatic progress between two and three million boards played per member, only to lose it as migrants invade and take over the population. Clearly the migrants exploited some weakness of the systems with low objective function score, but is is unclear without further investigation whether the systems performed well otherwise.

To test this, a system from one of those populations was reserved before being replaced (objective function = 5.597). Next, another system was similarly reserved from some time later (around 18 million boards per member) with an objective function score of 5.853. Having them compete over the 1,000 test board set, the second system beat the first by 901 IMPs, a large margin. While the objective function is a

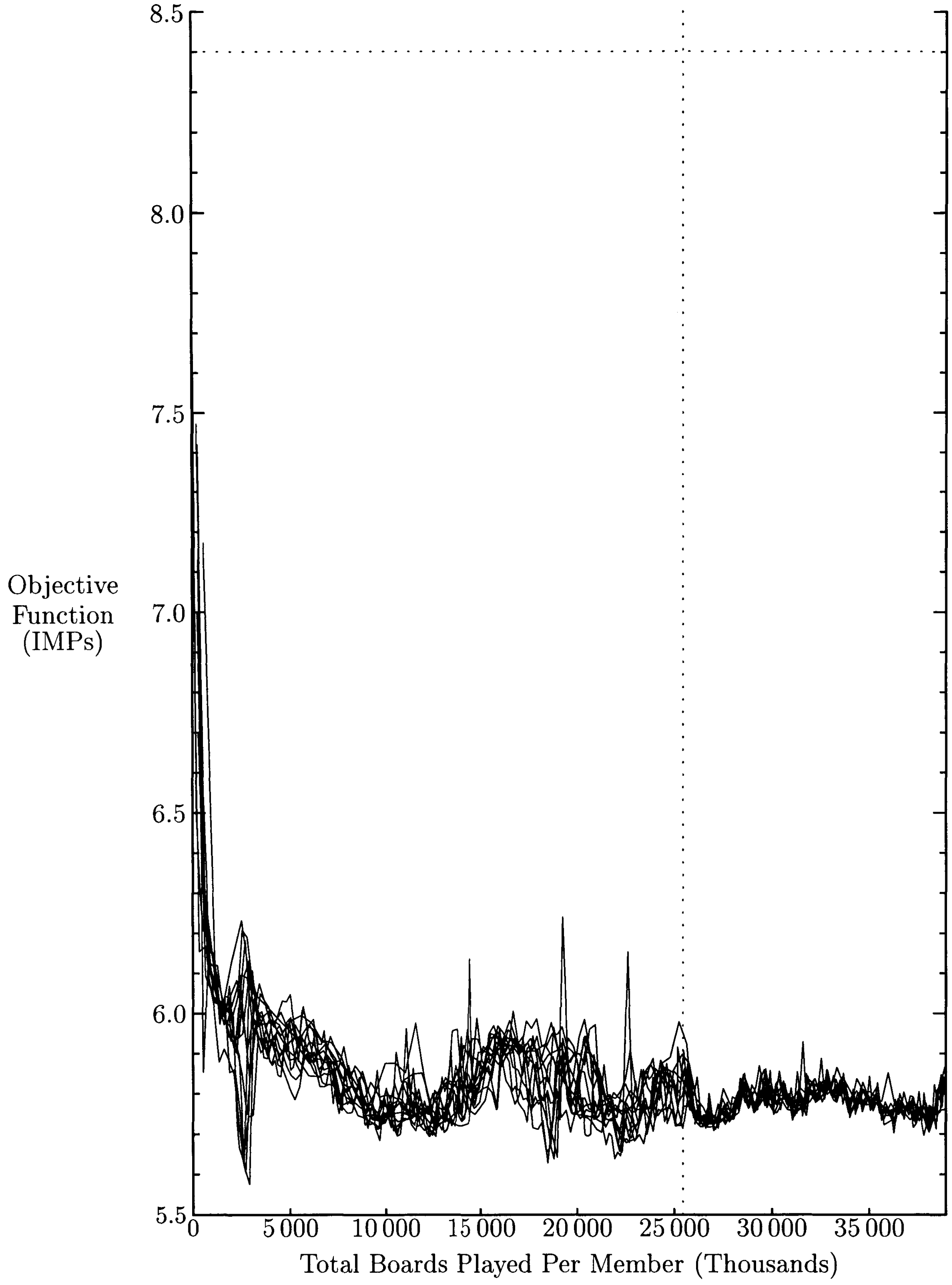


Figure 5-5: Run C (Multiple populations of Size 128)

function of performance, there are clearly other factors. A more detailed discussion of the objective function is presented in section 5.4.

Not surprisingly, when the migration rate was increased ten-fold (after 25.3 million boards played per member), the objective functions for the individual populations strayed less from each other, but no other effect was observed.

In two populations, systems reached an objective function score of 5.495 and 5.496 at 11 and 12 million boards per member, respectively. However, in each instance the systems had the best fitness for only one round-robin iteration. This supports the idea that systems with extreme objective function scores may not be strongest. Figure 5-6 demonstrates the performance of the 5.495 rated system.

Dlr: South	♠ 8762	
Vul: E/W	♡ QJ87	
	♢ Q	
	♣ QJ103	
♠ AK5		♠ 10
♡ 943		♡ A10652
♢ A85		♢ K10962
♣ 7542		♣ 96
	♠ QJ943	
	♡ K	
	♢ J743	
	♣ AK8	

West	North	East	South
			1♢[1]
Pass[2]	1♠[3]	Pass	Pass
Pass			

1♠ by North, making 3
N/S Score: 140 (Par = 140)

(1) 0–5 hearts and 13.0–16.2 HCP
(2) 0–16.2 HCP
(3) 0–10.0 HCP, usually 4+ spades

Figure 5-6: Run C example board

5.4 Objective Function Discussion

Up to this point, all references to objective function measurements have been with regards to WEBB systems. Additional insight may be gained, however, by applying the metric to human and other mechanical system bidding. To do this, the 40 boards of the 2001 OKBridge Worldwide Collegiate Internet Championship semi-finals (Harvard vs. Harvey Mudd) and finals (Harvard vs. Bilkent) were selected for comparison. Actual play results were ignored, using only the contracts and double-dummy results.

The hands were fed through the WEBB system discussed in Run C with a objective function score of 5.853, which nevertheless defeated a system with an objective function score of 5.597 by over 0.9 IMPs per board. Finally, they were also fed through GIB 3.3.0 playing two-over-one with one minute of thinking time on a Celeron 550 computer. The results are summarized in table 5.2.

	IMPs
pass-always	8.525
WEBB	6.475
Chen/Cotton	5.650
Sun/Woolever	5.350
GIB	4.425

Table 5.2: Objective function comparison

Interestingly, GIB obtained a very low objective function score. GIB chose to bid constructively in most instances, allowing the right contract to be found more often. On the other hand, the human players would often interfere boldly, an effective strategy against good opponents, but frequently ended up far from par because of it. Figure 5-7 illustrates this difference well.

As a result, to properly evaluate a bidding system, it may necessary to have systems compete directly against each other. This presents some complications, however, due to the need to explain bids. Because there is no universal way to describe a bid, it is a challenge to get GIB to understand bids made by WEBB and vice versa. This would be very desirable, however, and could be a goal of future work.

Dlr: East Vul: E/W	♠ AK10 ♡ A1092 ♢ 103 ♣ AQ106	
♠ Q8754 ♡ 73 ♢ Q4 ♣ KJ85		♠ 9 ♡ KQJ654 ♢ K8765 ♣ 4
	♠ J632 ♡ 8 ♢ AJ92 ♣ 9732	

West WEBB	North WEBB	East WEBB	South WEBB
		2♡[1]	Pass
Pass	2♠[2]	Pass	3NT[3]
Pass	Pass	Pass	

3NT by South, making 3
N/S Score: 400 (Par = 400)

(1) 5+ hearts and 0–14.0 HCP
(2) 0–4 hearts and 16.3+ HCP
(3) 5.8+ HCP

West GIB	North GIB	East GIB	South GIB
		Pass	Pass
Pass	1♣	2NT[1]	3♣
Pass	3♢	Pass	3♡
Pass	3NT	Pass	Pass
Pass			

3NT by North, making 3
N/S Score: 400 (Par = 400)

(1) Shows hearts and diamonds

West Chen	North Celikler	East Cotton	South Kececioglu
		4♡	Pass
Pass	Double	Pass	4♠
Pass	Pass	Pass	

4♠ by South, down 1
N/S Score: -50 (Par = 400)

West Deniz	North Sun	East Bekir	South Woolever
		2♡	Pass
Pass	2NT	Pass	Pass
Pass			

2NT by South, making 3*
N/S Score: 150 (Par = 400)

* making 2 (120) in actual play

Figure 5-7: Player comparison

Chapter 6

Suggestions for Future Research

The results obtained by WEBB are impressive considering how simple it is compared to other bidding engines. As a result, there are many ways it can be improved. Some are explored in this chapter.

6.1 Competitive Bidding

Currently, while WEBB uses information that can be derived from bids, it does not use the bid itself. In order to be competent at competitive bidding, mechanisms would need to be added to observe and react to opponents bids.

6.2 Advanced Hand Evaluation

While HCP is a useful metric for hand evaluation, it is not sufficient for advanced bidding. Adjustments to this metric, similar to how COBRA computes BPC and DPC, may provide a significant boost.

6.3 Uncertain information

While FIA and GIA are well-suited for dealing with static information, they are not easily adapted to process features that may change value or with processing probabilistic information. Without these capabilities, WEBB is fundamentally limited.

6.4 High-level decision nodetypes

While strength and suit-length decision nodetypes provide basic functionality, good bidding systems would incorporate other nodes operating on the state of bidding, keycards, shape, stoppers, suit quality, and vulnerability. An alternative would be to provide a mechanism for allowing WEBB to devise its own decision nodetypes rather than just adjusting parameters to given nodetypes.

6.5 Expert Agents

While decision graphs are an effective way to choose a bid, as presented they lack a way to express higher-level thinking. By creating expert agents for specific problems and having their findings available to the overall decision making process, a significant performance boost can be realized. As a side effect of the dynamic effects these agents will produce, FIA and GIA will fail, however.

6.5.1 Contract Evaluation

The most basic of which is an agent that evaluates different contracts. Björn Gambäck and Manny Rayner have done work using neural networks to estimate contract success probabilities [5]. It is possible to take this further and to extend the agent to list uncertain features that would best determine the appropriate contract. Knowing this, the decision process could try to determine the most valuable information needed at any given time and direct the auction appropriately.

6.5.2 Preempt Evaluation

To effectively preempt, a bidding system needs to consider the potential penalty that the opponents may inflict, the likelihood that the penalty will be extracted, how effective the preempt will be, and how likely the loss of bidding space will hinder its partner. Incorporating these into an expert agent, it could advise the decision process whether or not a preempt is appropriate, and at what level to do so.

6.5.3 Save Evaluation

Similar to preempt evaluation, if the opponents bid game or slam, a high-level process should determine whether a save is available and consider it versus the probability that the opponents will make their contract.

6.5.4 Penalty Evaluation

Finally, if the opponents make a bid that will probably not be made, some consideration should be given to penalizing them. In conjunction with the contract evaluator, the penalty evaluator could estimate the probability the opponents may make a contract, the benefit of penalizing them, the cost of doubling a making contract, and whether playing another contract may produce a higher score.

6.6 Comparison with COBRA

If some of the above features were added to make WEBB a better bidder, it would be interesting to see how it compares to COBRA in terms of both system behavior and performance.

6.7 Human Utility

If good bidding systems can be generated, there will be thousands of Bridge players eager to examine the inner workings of such systems. Unfortunately, evolutionary systems are notorious for producing cryptic solutions escaping analysis and human understanding. Presumably, the best way a human could understand such a system is by example. Given a bidding sequence, the bidding system should be able to describe what all continuations mean.

An alternative is to require that the only information that can be expressed with bids be expressible succinctly in English or some other natural language. While this may be slightly restrictive, systems that cannot be described effectively are useful only as a Bridge curiosity.

Chapter 7

Summary

WEBB is a modular, extensible program with the goal of producing improving bidding systems for Bridge. It is a fusion of two inter-working components: expert knowledge of bridge and a co-evolutionary competitive learning module. Expert knowledge is expressed in WEBB in terms of hand features and bid description. Other knowledge could be easily added in form of additional hand features to improve performance, but the purpose of this work is to demonstrate that evolutionary learning can be effectively applied to the domain of Bridge bidding. Co-evolution is essential for the learning module to operate effectively, using IMP totals as a fitness measure to guide genetic reproduction, inheritance, and variation.

WEBB is not only evaluated in terms of how bidding systems improve, but also with an external objective measure of bidding a 1,000-board evaluation set. Unfortunately, neither the objective function nor competition are completely accurate in assessing true ability. However, by both metrics, WEBB obtained fair results, despite its basic operator set. This provides indications that performance-based improvements could have a significant effect, as WEBB adapts to incorporate the new information into bidding systems.

Appendix A

Glossary

auction a sequence of bids, ultimately leading to a contract.

board a bridge deal, usually played multiple times by different people

COBRA *C*omputer *O*riented *BR*idge *A*nalysis – a computer optimized bidding system designed by E.T. Lindelöf [9]

contract the result of bidding, including a trump suit, level, and whether or not it is played [re]doubled (Ex. 3NT-X)

double-dummy This refers to each player being aware of what specific cards each other player has. This perfect information is never available in real games, but it is a useful approximation of actual conditions and is much easier to evaluate.

double jump-shift a bid of some suit (or no-trump) when the bidder has the option of bidding the same suit (or no-trump) at two levels lower (ex. 1♠–4♡)

FIA Feature Inferencing Algorithm (section 3.4.1)

free bid a bid made after the previous play has made a bid, meant to indicate the case where a player's partner would have an opportunity to bid whether or not the player bids.

GIA Graph Interference Algorithm (section 3.4.2)

GIB Ginsberg's Intelligent Bridgeplayer

HCP high-card points

high-card points a simple hand evaluation metric which gives weight to cards of different ranks which are summed together to form the HCP total for a hand

IMP a type of scoring where the point difference is scaled approximately logarithmically (refer to table B.1)

jump-shift a bid of some suit (or no-trump) when the bidder has the option of bidding the same suit (or no-trump) at one lower level (Ex. 1♡–2♠).

LHO left-hand opponent

par the expected score on a deal

pard partner

preempt a high-level bid showing a weak unbalanced hand (often with one long suit) for the purpose of disrupting the opponents' bidding

push using IMP scoring, a raw score difference or -10, 0, or +10 which corresponds to a zero IMP score

RHO right-hand opponent

save a contract not meant to be made, but bid on the belief that even if doubled, the penalty will be less than what the opponents can score if allowed to play the hand

two-over-one a common bidding system

WEBB Woolever's Evolving Bridge Bidder

Appendix B

Introduction to Duplicate Bridge

Duplicate Bridge is a card game played with at least eight players. The most common arrangement, involving two teams of four players, involves two players from each team sitting north-south at one of two tables, with their teammates sitting east-west versus their opponents at the other table. Each deal of bridge played (called a *board*) is played at each table. By having each team play each board in each direction, a net score can be generated based on how well each team does compared to the other. This net score is converted to International Matchpoints (IMPs) based on table B.1.

ΔPoints	IMPs	ΔPoints	IMPs	ΔPoints	IMPs
20–40	1	370–420	9	1500–1740	17
50–80	2	430–490	10	1750–1990	18
90–120	3	500–590	11	2000–2240	19
130–160	4	600–740	12	2250–2490	20
170–210	5	750–890	13	2500–2990	21
220–260	6	900–1090	14	3000–3490	22
270–310	7	1100–1290	15	3500–3990	23
320–360	8	1300–1490	16	4000+	24

Table B.1: International Matchpoint scale [1]

B.1 Mechanics

A standard deck of 52 cards is shuffled and divided equally among four players. One player is considered the dealer (which rotates with each hand) who has the opportunity to make the first bid. A bid consists of a number, one to seven, and a suit (or no-trump). The number indicates how many tricks the bidder will try to take beyond six if the chosen suit is trump. A one-level contract indicates an attempt at a simple majority of the tricks and a seven-level contract indicates an attempt to take all thirteen tricks. Instead of choosing to bid any player may pass, double a bid of their opponents, or redouble an opposing double. Bidding proceeds clockwise until three people pass in sequence (or four if noone has yet opened the bidding). Each bid must be greater than the previous bid in count, or equal in count but higher in suit (ranked from lowest to highest: clubs, diamonds, hearts, spades, no-trump).

If all four players pass, the score for each partnership is zero. Otherwise play commences, with the side that chose the last bid (known as the *contract*) trying to take at least as many tricks as indicated by the contract while their opponents try to prevent this. Of the two players on the partnership playing the hand, the player who first suggested playing in the trump suit is the *declarer.* The opponent to the left of the declarer chooses a lead after which the declarer's partner (the *dummy*) spreads his cards on the table. The declarer chooses a card to play from these, the other opponent plays a card, and finally the declarer plays a card to constitute a *trick.* Each player must follow with the suit led, if possible. The player with the highest ranking card of the led suit (or the highest ranking trump if any are played) wins the trick and leads a card starting the next trick. This process repeats until all thirteen tricks have been claimed.

B.2 Score

For each board, either or both sides may be *vulnerable* which increases the bonuses for bidding and making *game* or *slam.* However, the penalty for failing to take the indicated number of tricks also increases. Each partnership earns the additive inverse of their opponents' score.

B.2.1 Contract is made

20 points are awarded for each trick claimed and made with clubs or diamonds as trump (30 points otherwise). If the contract is no-trump, 10 additional points are added. If the contract is doubled or redoubled, this score is doubled or redoubled, accordingly. If the result is 100 or more points, the contract is called a *game contract*, and an additional bonus is available (refer to table B.2). Otherwise 50 points are awarded for making a contract. If twelve tricks are claimed and won, the contract is called a small slam. Finally, if all thirteen tricks are claimed and won, the contract is referred to as a grand slam (refer to table B.3).

	♣/♢	♡/♠	No-trump
Undoubled	5	4	3
Doubled	3	2	2
Redoubled	2	1	1

Table B.2: Minimum contract levels needed for game bonus

	Not vulnerable	Vulnerable
Part score – or – Game bonus	50 – or – 300	50 – or – 500
Small slam	500	750
Grand slam	1000	1500

Table B.3: Contract bonuses[1]

Additional tricks beyond those required for the contract are worth 20 points each if clubs or diamonds is trump (30 points otherwise). If the contract is doubled, additional tricks are worth 100 or 200 instead (regardless of the trump suit). If the contract is redoubled, additional tricks are worth 200 or 400 instead. Finally a bonus of 50 points is added for making any doubled contract or 100 points for any redoubled contract.

	Not vulnerable	Vulnerable
Undoubled	normal value	normal value
Doubled	100	200
Redoubled	200	400

Table B.4: Overtrick bonuses[1]

B.2.2 Contract is set

If the opponents are successful at setting the contract, they earn 50 or 100 points per trick, depending on the vulnerability of the declaring side. If the contract is doubled, the award increases to 100 points for the first trick under the count needed, 200 points for each of the second and third tricks, and 300 points for any subsequent tricks. If declaring side is vulnerable, instead use 200 points for the first trick and 300 points for subsequent tricks. If the contract is redoubled, the award is doubled.

	Not vulnerable				Vulnerable			
	1	2	3	4+	1	2	3	4+
Undoubled	50	100	150	+50/each	100	200	300	+100/each
Doubled	100	300	500	+300/each	200	500	800	+300/each
Redoubled	200	600	1000	+600/each	400	1000	1600	+600/each

Table B.5: Undertrick penalties[1]

Bibliography

[1] American Contract Bridge League (ACBL). Laws of duplicate contract Bridge. http://www.acbl.org/info/laws97/index.htm, May 1997.

[2] Michael Buro. Takeshi Murakami vs. Logistello. http://www.neci.nj.nec.com /homepages/mic/ps/match-report.ps.gz, August 1997.

[3] Gay Carley. A program to play contract Bridge. Master's thesis, Dept. of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 1962.

[4] Ian Frank. Computer Bridge survey. Technical Report TR-97-3, ETL, February 1997.

[5] Björn Gambäck and Manny Rayner. Contract Bridge as a micro-world for reasoning about communication agents. Technical Report SICS/R-90/9011, Swedish Institute of Computer Science, 1990.

[6] Gibware.com. GIB. http://www.gibware.com.

[7] Matthew L. Ginsberg. GIB: Steps toward an expert-level Bridge-playing program. ftp://ftp.cirl.uoregon.edu/pub/users/-ginsberg/papers/gib.ps.gz, 1998.

[8] IBM. Kasparov vs. Deep Blue: The Rematch. http://www.research.ibm.com/ deepblue/home/html/b.html, May 1997.

[9] E.T. Lindelöf. COBRA*: The Computer-Designed Bidding System.* Victor Gollancz Ltd, London, 1983.

[10] Department of Computer Science University of Alberta. Chinook. http://www.cs.ualberta.ca/~chinook, August 1998.

[11] A. Wasserman. Realisation of a skillful Bridge bidding program. In *Proceedings of the Fall Joint Computer Conference — AFIPS*, Houston, Texas, 1970.

5701-23